Forecasting without context problem

José Ortiz-Bejar, Jesús Ortiz-Bejar,
Alejandro Zamora-Mendez
Universidad Michoacana de San Nicolás de Hidalgo
Michoacán, México
{jortiz,jesus.ortiz,azamoram}@umich.mx

Garibaldi Pineda-García
University of Sussex
Brighton, United Kingdom
g.pineda-garcia@sussex.ac.uk

Mario Graff, Eric S. Tellez
CONACyT - INFOTEC
Aguascalientes, México
{mario.graff,eric.tellez}@infotec.mx

Abstract

This work presents an analysis of four regression systems. Two of them are statistical: the widely used Auto-regressive Integrated Moving Average (ARIMA) and the state-of-the-art Facebook Prophet. From the deep learning school, a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) is also evaluated. We finish our quartet with a fine-tuned Nearest Neighbor model. The study is carried out over seventeen benchmarks; fifteen coming from M4-Competition and two more power systems time series, i.e., electricity demand and hydropower generation. For all the models, the regression systems are fitted and optimized to minimize user intervention. The results show that deep learning models obtained the best performance; nonetheless, the performance difference is not statistically significant with the rest of the systems tested.

1 Introduction

In many scientific fields, it is common to model processes or phenomena as a sequence of states or measurements. For instance, the Bitcoin price variation, load evolution in a power system, network traffic, and many others. Formally, the set of all possible states is denominated state space. It is unfeasible for many processes and phenomena to obtain a complete state-space or an equivalent mathematical expression; only a finite sequence of states

that were measured, directly or indirectly, are available. This finite sequence sample from the full state space is formally called time series. The problem of reconstructing the full state space from a given time series relies mainly on the Takens's embedding theorem [1]. Takens's theorem states that from a given time series $X = \{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$, it is possible to generate a model for the space state U. More detailed, for a sub-sequence (window) of observations w of dimension m (embedding dimension) and a constant time delay τ, there exists a function f such that:

$$x(t) = f(w) = f\left[x_{(t-\tau)}, x_{(t-2\tau)}, \ldots, x_{(t-(m-1)\tau)}\right] \tag{1}$$

By knowing the model in Equation 1, it is possible to predict the state at any time t by using m previous observations sampled at τ intervals. Unfortunately, function f is rarely available, or it is impossible to determine analytically.

Note that deep learning is a branch of machine learning dealing with neural networks with several hidden layers. Therefore, it is necessary to find the correct modeling for the time series, i.e., the best values for m and τ. Usually, determining the correct values for m and τ is hard work and requires domain expertise.

Our contribution is a study about the performance of four regressors in seventeen well-known time-series. Among the four methods tested, we included our previous work, KNNRS, a minimalist regressor based on nearest neighbors and finely tuned with a random search on a unique configuration space that includes both m, τ, and the parameters of the nearest neighbor regressor, detailed in §2.2. We found that our method is competitive with more sophisticated alternatives under our benchmarks, being statistically similar to recursive neural networks without major hardware requirements or computing frameworks.

The rest of this manuscript is as follows. Section 2 reviews briefly the related work in the literature, and in particular, it describes in-depth the four regressors studied. Section 3 is dedicated to our experimental methodology and result presentation. Finally, our results are discussed in §4.

2 Related Work

The literature for time series modeling is vast. Many popular techniques can be seen as finding optimal values for parameters m and τ dates. For instance, several techniques define criteria and methodologies to find m and τ independently. One of the simplest criteria to calculate τ was proposed by **(author?)** [2], in which $\Phi(\tau)$ is defined as the auto-correlation function for the time series and the τ value is set as $\Phi(\tau) = 1/e$ where e is the statistical error on estimated dimension. One common way to determine the value for m is by using the *False Nearest Neighbor* algorithm [3]. This method

evaluates a subset of the nearest neighbors as a function of the embedding dimension. The minimum value for m is found when a large number of nearest neighbors do not separate by more than a predetermined threshold as the dimension is increased. One of the main disadvantages of the *False Neighbor* algorithm is that it is highly dependent on the threshold value. A method based on Differential Entropy to determine m and τ simultaneously was defined by **(author?)** [4]. A detailed analysis of the importance of how the values m and τ impact the forecasting can be found in the works of **(author?)** [5], and **(author?)** [6].

2.1 Auto ARIMA

Traditionally, time series forecasting is performed using statistical methods, the most popular being the Auto-Regressive Integrated Moving Average (ARIMA), ARMA, and AR [7]. These methods use statistical measurements and regression terms to create patterns that allow predicting future outcomes. It is assumed that it is possible to forecast future values based on past events, like Takens's theorem. ARIMA's prediction strategy is mainly based on lags and forecast errors. The predicted value for a x_t is a weighted sum of one or more previous values (named auto-regressive terms) and recent prediction errors. Roughly, ARIMA's hyper-parameters are the number of previous observations p, the number of non-seasonal differences d, and the number of lagged forecasting error items.

An automated tool to optimize AR family models is the widely use Auto ARIMA [8]; there are implementations available for the R and Python languages. Auto ARIMA implements a variation of the Hyndman-Khandakar algorithm [8], which combines unit root tests, minimizes the Akaike information criteria, and Maximum Likelihood Estimation to determine ARIMA hyper-parameters. To use ARIMA, it may be necessary a profound understanding of the problem under study.

2.2 KNN Approaches

Since determining m and τ is a key issue, KNN approaches focus on optimizing their values. The simplest method is the deterministic Nearest Neighbor [9], which computes the parameters using the False Neighbor algorithm and the Mutual Information algorithm. This strategy is considered deterministic because it uses a fixed neighborhood radius of $\epsilon = 1 \times 10^{-3}$ and updating $\epsilon = 1.2\epsilon$ when no nearest neighbors are found. Researchers have used Differential Evolution (DE) to optimize parameters m, τ, and ϵ simultaneously; this strategy allows us to build accurate models for time series forecasting [10, 11]. Even though DE is relatively simple, multiple parameters need to be defined, i.e., boundaries for m, τ and ϵ, population size, scale factor, recombination probability, the maximum number of itera-

tions, and the number of executions. **(author?)** introduced a Fuzzy version of the kNN algorithm (FNN), their approach does not include m, τ and k/ϵ optimization. Moreover, it is necessary to generate a set of linguistic terms and prediction rules.

kNN Random Search (KNNRS) optimizes the values for m, τ, and the parameters for a kNN regression systems the distance function d, number of neighbors k, and the neighbor weighting scheme ω by using the Random Search (RS) heuristic [13]. Only a small subset of the possible models is evaluated through random sampling of the configuration space; the best model is selected through cross-validation schemes. The configuration space describes a large set of combinations of ($k,\ \tau,\ m,\ d,\ \omega$), that is, the KNNRS regressor selects all parameters jointly. We use KNNRS in our comparative study.

2.3 Facebook Prophet

Thinks of time series as decomposable with *trend* (non-periodic; g), *seasonality* (s), and *holiday* components (h). The approach is to do curve fitting of the different components and combine them to produce a complex time series. For the *trend* component, the Prophet model uses either a logistic function with an adjustable rate or a linear piece-wise function. The *seasonality* component is modeled using the Fourier series to capture a range of frequencies of periodic events. Most business time-series are affected by human activities, which may not be periodic; the *holiday* component allows the time series to include such events. This methodology offers fully automatic regression and the possibility for the user to impose some of its knowledge on the system [14].

2.4 LSTM

Typical Artificial Neural Networks use feed-forward connectivity; information flows from input to output in a single direction. In contrast, Recurrent Neural Networks (RNNs) have feedback connections that create loops through which information can remain in the network. The feedback connections in RNNs make them a natural fit to analyze temporal data, such as time series.

A major issue with RNNs are long-term dependencies have little effect while training with gradient descent [15]. Long-short Term Memory (LSTM) networks [16] are a type of recurrent artificial neural network with gated recursion to control information flow, and they have been designed to tackle the long-term dependency problem. Recent advances in computer hardware have allowed the training of sufficiently large LSTMs, which, in turn, have been included in commercial products for voice recognition, for example.

Like most neural networks, LSTMs are trained using gradient descent as

Table 1: Benchmark datasets

M4id/Name	Category	Sampling rate	Horizon	Dataset size
D162	Micro	Daily	14	4411
D447	Micro	Daily	14	4555
D448	Micro	Daily	14	4555
D449	Micro	Daily	14	4457
D450	Micro	Daily	14	4440
D451	Micro	Daily	14	4440
D454	Micro	Daily	14	4440
D457	Micro	Daily	14	4412
D588	Micro	Daily	14	4740
D1612	Demographic	Daily	14	4440
D1613	Demographic	Daily	14	4440
D2015	Industry	Daily	14	4555
D2047	Finance	Daily	14	8519
D4099	Other	Daily	14	9919
D4226	Other	Daily	14	4440
HG	Power Systems	Hourly	168	8016
ED	Power Systems	Hourly	168	8016

a (self) supervised problem. Typically, a data block is chosen as the input to predict another block (the target). Like other approaches, window size selection is not trivial and requires knowledge of the time series' nature. Additionally, most common hyper-parameters (e.g., learning rate, regularization, or optimization algorithms) selection issues associated with deep neural networks are present for LSTMs.

3 Experiments and results

To evaluate each of the forecasters, a set of fifteen time-series taken from the M4-Competition [17] and two power systems times series for *Electricity demand (ED)* and *Hydropower Generation (HG)* [18]. Table 1 summarizes the characteristics for each evaluated dataset. All datasets from the M4-Competition are sampled at a daily frequency. Please note that the prediction horizon for the M4 time series is 14 days, while the power systems dataset is seven days, which is equivalent to 168 hours.

All the M4 time series were selected with more than 4000 samples, and with no zero values, the latter allows us to use the Median Absolute Percent Error (*MAPE*) as a fitness function, which is defined by Eq. 2.

$$MAPE(x, x_p) = 100\frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - x_{p_i}}{x_i}\right|, \tag{2}$$

where x_i is the actual value for the time series at time $t = i$ and x_{p_i} is the predicted outcome. *MAPE* is used because it is easy to interpret, scale-independent, and one of the most widely used to measure regression systems performance.

3.1 Forecasting strategy

Each quality measurement function is evaluated by using N (see *horizon* column at Table 1) predicted values. Before defining our experimental setup, we briefly explain how the prediction process is performed. The N values are predicted by applying N steps ahead (NSA) strategy. To compute the value $x_{p_{t+1}}$ (i.e the value at time $t+1$) models use a window of dimension m with values from $t-(m-1)\tau+1$ to $t-\tau+1$ (see Eq. 3); then the predicted value is appended to the time series and will be used to compute future values. Please notice that for any $t+i$ where $\tau < i < m\tau$ a mixing of true values and predicted values are used by regression system, while for $i > m\tau$ the input window contains only model predictions.

$$x_{p_{t+i}} = f\left[x_{(t-\tau+i)}, x_{(t-2\tau+i)}, \ldots, x_{(t-(m-1)\tau+i)}\right] \tag{3}$$

3.2 Experimental setup

For each of the M4 benchmark datasets, the training set X is used to fit each model, and its corresponding test set Y to measure regression systems performance. For LSTM and KNNRS, the last 14 samples (days) of the training sets are used as a validation set. Both KNNRS and LSTM models are optimized using MAPE as fitness (see Eq. 2).

On the other hand, for power systems time series, the 8016 samples (i.e., eleven months) are used as a training set X during the model optimization phase, and the last month (744 samples) test set Y to measure regression systems performance. LSTM and RSNN use the last 168 samples (7 days) of the training set as a validation set. In this case, MAPE (see Eq. 2) is again used as a fitness function for LTSM and KNNRS.

Two quality measurements are applied to evaluate forecasters' performance: the previously mentioned *MAPE* and the maximum residual error (*Max*), which is reckoned using Eq. 4. The *Max* metric captures the worst-case error between the predicted value and the actual value.

$$Max(x, x_p) = \max|x_i - x_{p_i}| \tag{4}$$

Auto ARIMA, Prophet, and KNNRS are optimized with no user configuration only the training set X is given. However for the LSTM the values for m and τ are manually defined as $m = 4$ and $\tau = 7 \times 24$ for power systems time series and $m = 4$ and $\tau = 7$ for all the M4 problems. Those values were defined under the assumption that all the involved time series depend on past observation happening on the same day/hour.

3.3 Results

First, we analyze qualitatively M4 and power systems time series independently, then we proceed to summarize the results by an average rank based on *MAPE*.

3.3.1 M4 time series

Figure 1 shows the error distribution for each one of the problems taken from the M4 competition. As can be seen, Prophet has the lowest performance; its median error is around 10% and has the most significant variance of the four evaluated systems. On the other hand, *LSTM, KNNRS* and *Auto ARIMA* exhibit a similar median value (around of 5%). *Auto ARIMA* has the lower variance and *KNNRS* a slightly less median value.

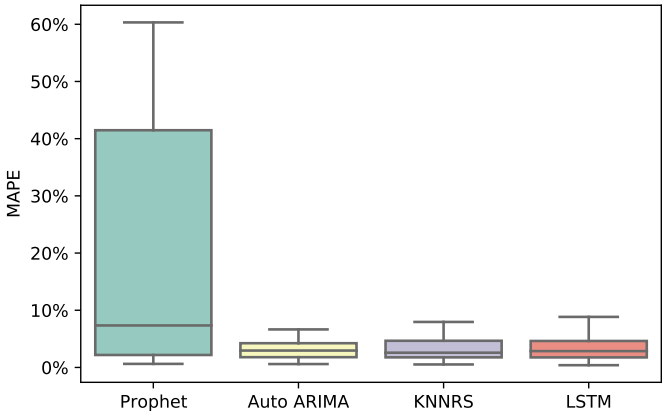

Figure 1: Boxplot of *MAPE* distribution for M4 benchmarks

In Figure 2 the distribution for *Max* Error for M4 data is shown. Form Figure 2 can be appreciated that Prophet is the model with the higher error values, while the other three approaches having similar error distribution. Please recall that *Max* error is measured at the time series scale and not in percent. The *Max* error is relevant in situations where a huge difference in forecasting can be lead to a catastrophic event (e.g., bankruptcy, voltage collapse, or even to death).

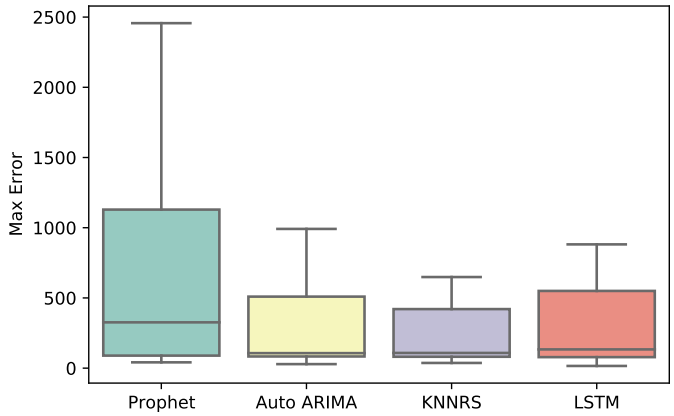

Figure 2: Boxplot of *Max* error distribution for M4 benchmarks

3.3.2 Power systems time series

For the power systems datasets, the optimized models are used to predict different number of steps ahead for $N \in \{24, 48, 72, 86, 144, 168, 336, 720\}$. The previous scenario may aid in planning energy dispatch, system growing, maintenance schedule, among others. Figure 3 shows that *Auto ARIMA* has the worst performance having an error around of 20% for the HG time series and more than 15% for ED. For the remaining forecasters, KNNRS has the best median value for the HG time series and the ED's minimum variance. While Prophet exhibits the best performance for ED, and it is the second-best for HG. LSTM has similar performance to Prophet and KNNRS for the HG time series and an inferior performance for the ED.

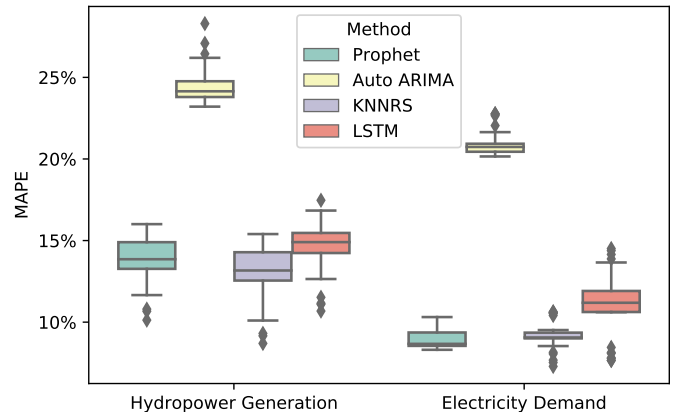

Figure 3: Boxplot of *MAPE* distribution for Power Systems time series

Figure 4 illustrates the distribution of *Max* error for the different prediction lengths. According to Figure 4, Prophet has the lower median MAPE

for power systems problems, and the one with the minimal variance is KNNRS. However, all forecasters exhibit extreme worst cases for both time series.

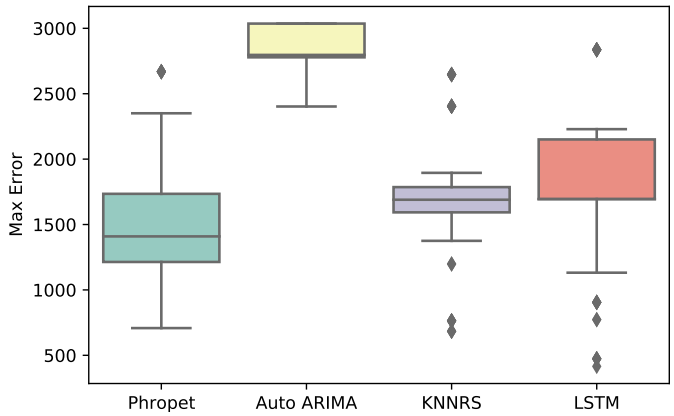

Figure 4: Boxplot of *Max* error distribution for Power Systems time series

To illustrate how the model's prediction evolves as the prediction size grows, we have included it in Figure 5. The first 24 hours for a one-month prediction size are shown at the top, while the bottom part shows the last 24 hours for the same period; both plots illustrate the ED time series. It can be appreciated that for the first 24 hours Prophet, KNNRS, and LSTM produce curves that are pretty similar to the real value, while Auto ARIMA only does so at the beginning. On the other hand, for the last 24 hours, all the models perform poorly; however, while Prophet and KNNRS keep the signal shape, LSTM exhibits at flat behavior near the actual max value, and Auto ARIMA is fixed at the average.

3.3.3 Average Rank

In this section, a quantitative analysis based on each approach's average rank with each time series is carried out. The regression systems are ranked using the MAPE score, and results are summarized in Table 2. For instance, the problem *D162* the LSTM gets the first place due to it has the lowest MAPE, KNNRS is the second best, Auto ARIMA the third, and Prophet has the lower performance and got the fourth place. To ease reading in Table 2, the best values are in boldface, and the rank is indicated as a subscript. From Table 2, it is clear that, based on the number of first places (10 out of 17) and the average rank, LSTM outperforms the other three systems in the evaluated benchmarks. KNNRS has the second-best average rank, even when it has only two best places, as it never gets any last place. Prophet has the third-best rank. It is worth mentioning that Prophet has the most erratic performance since it is a model with more fourth places (9)

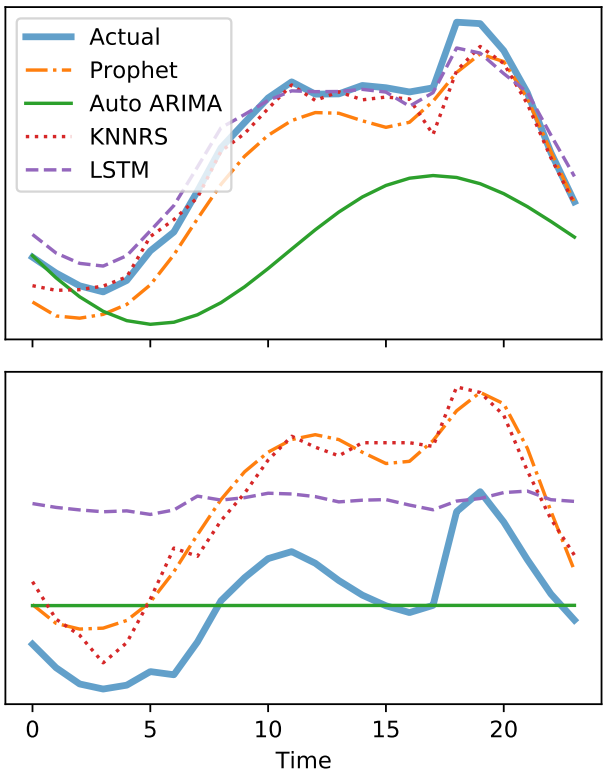

Figure 5: Predictions for the first (top) and last (bottom) 24 hours for a period of one month

Table 2: Average Rank by MAPE score for the seventeen benchmark datasets

Time Series	LSTM	KNNRS	Prophet	Auto ARIMA
D162	**1.929**$_{(1)}$	2.252$_{(2)}$	8.745$_{(4)}$	2.294$_{(3)}$
D447	**3.278**$_{(1)}$	7.956$_{(3)}$	76.655$_{(4)}$	6.649$_{(2)}$
D448	3.127$_{(2)}$	5.211$_{(3)}$	**1.875**$_{(1)}$	8.422$_{(4)}$
D449	**8.837**$_{(1)}$	11.710$_{(3)}$	59.814$_{(4)}$	10.278$_{(2)}$
D450	**1.884**$_{(1)}$	3.335$_{(3)}$	58.640$_{(4)}$	3.203$_{(2)}$
D451	**1.627**$_{(1)}$	4.729$_{(3)}$	60.325$_{(4)}$	3.189$_{(2)}$
D454	5.840$_{(4)}$	2.149$_{(3)}$	**1.514**$_{(1)}$	2.125$_{(2)}$
D457	**2.858**$_{(1)}$	4.600$_{(2)}$	24.308$_{(4)}$	4.837$_{(3)}$
D588	5.811$_{(4)}$	**0.632**$_{(1)}$	5.158$_{(3)}$	0.640$_{(2)}$
D1613	2.826$_{(3)}$	2.578$_{(2)}$	**1.365**$_{(1)}$	2.935$_{(4)}$
D2015	**0.519**$_{(1)}$	0.523$_{(2)}$	7.342$_{(4)}$	0.593$_{(3)}$
D2047	**0.567**$_{(1)}$	1.379$_{(3)}$	11.536$_{(4)}$	0.787$_{(2)}$
D4099	3.448$_{(2)}$	3.585$_{(3)}$	**2.481**$_{(1)}$	3.637$_{(4)}$
D4226	**0.395**$_{(1)}$	0.699$_{(2)}$	4.990$_{(4)}$	1.458$_{(3)}$
ED	**3.452**$_{(1)}$	3.768$_{(3)}$	3.568$_{(2)}$	15.883$_{(4)}$
HG	9.707$_{(3)}$	**7.289**$_{(1)}$	8.312$_{(2)}$	19.524$_{(4)}$
Avg. Rank	**1.882**$_{(1)}$	2.411$_{(2)}$	2.823$_{(3)}$	2.882$_{(4)}$

and, at the same time, the second in the number of best places. Finally, Auto ARIMA has a lower rank, but it is essential to mention that it is the model with more second places.

A Wilcoxon signed-rank test over MAPE performances shows that LTSM is statistically similar to KNNRS, Prophet, and Auto ARIMA with p-values of 0.45, 0.05, 0.07, respectively. KNNRS is statistically similar to Auto ARIMA with a p-value of 0.37 but statistically different from Prophet with a p-value of 0.01. Auto ARIMA and Prophet are statistically equivalent, having a p-value of 0.09.

Then, an average rank base in *Max* metric is shown in Table 3. In this case, the results are similar to those based in *Max*; since LSTMs are the first ranked, and KNNRS the second one. However, places between Auto ARIMA and Prophet are exchanged. A Wilcoxon signed-rank based on *Max* shows that LSTM is statistically similar to KNNRS and Auto ARIMA with p-values of 0.2 and 0.07, respectively, and better than Prophet with a p-value of 0.03. KNNRS is similar to both Prophet and Auto ARIMA with p-values of 0.05 and 0.42, respectively. Finally, Prophet and Auto ARIMA are statistically similar, with a p-value=0.2.

	LSTM	KNNRS	Auto ARIMA	Prophet
D162	**881.37**$_{(1)}$	979.46$_{(2)}$	991.50$_{(3)}$	2456.62$_{(4)}$
D447	**618.45**$_{(1)}$	956.77$_{(3)}$	804.75$_{(2)}$	10883.56$_{(4)}$
D448	**481.37**$_{(1)}$	648.77$_{(3)}$	953.78$_{(4)}$	551.09$_{(2)}$
D449	**1664.05**$_{(1)}$	2338.90$_{(3)}$	1781.90$_{(2)}$	6910.60$_{(4)}$
D450	**31.08**$_{(1)}$	38.02$_{(2)}$	38.02$_{(2)}$	325.54$_{(4)}$
D451	**15.65**$_{(1)}$	39.71$_{(3)}$	37.07$_{(2)}$	326.12$_{(4)}$
D454	233.00$_{(4)}$	102.53$_{(2)}$	102.53$_{(2)}$	**95.36**$_{(1)}$
D457	88.45$_{(3)}$	**85.97**$_{(1)}$	88.35$_{(2)}$	317.47$_{(4)}$
D588	690.21$_{(4)}$	113.42$_{(2)}$	**106.86**$_{(1)}$	551.50$_{(3)}$
D1613	121.54$_{(4)}$	108.26$_{(2)}$	117.81$_{(3)}$	**72.04**$_{(1)}$
D2015	**133.50**$_{(1)}$	192.25$_{(2)}$	213.91$_{(3)}$	1143.86$_{(4)}$
D2047	**91.69**$_{(1)}$	191.88$_{(3)}$	115.96$_{(2)}$	1114.31$_{(4)}$
D4099	68.34$_{(2)}$	76.15$_{(3)}$	77.86$_{(4)}$	**48.92**$_{(1)}$
D4226	**15.94**$_{(1)}$	37.01$_{(3)}$	28.65$_{(2)}$	83.10$_{(4)}$
ED	**772.72**$_{(1)}$	1375.19$_{(3)}$	2896.65$_{(4)}$	862.23$_{(2)}$
HG	1221.96$_{(3)}$	1198.25$_{(2)}$	2778.25$_{(4)}$	**1069.29**$_{(1)}$
Avg. Rank	**2.00**$_{(1)}$	2.41$_{(2)}$	2.64$_{(3)}$	2.82$_{(4)}$

Table 3: Average Rank by *Max* score for the seventeen benchmark datasets

4 Conclusion

In this work, we presented a qualitative and quantitative analysis of state-of-the-art regression systems. From the qualitative and quantitative results, it may be inferred that Prophet exhibited a lower accuracy. Simultaneously, the other three may be considered similar when no advantage is taken from the problem context. Even though LSTM shows the best performance, it was necessary to define the values for m and τ hyper-parameters. KNNRS tends to be in the middle of the table, being ranked second or third in most experiments. Facebook Prophet shows the most erratic behavior since most of the time is at first or the last one. Auto ARIMA is at third and fourth places on the Average Rank. It has only lousy performance for power systems times series but similar to the other regressors for the M4 datasets. However, the Wilcoxon signed-rank shows that only Prophet and KNNRS models are statistically different, with more than 98% confidence.

Though its performance is not consistent throughout our tests, Prophet is fast to train and evaluate, allowing a user-in-the-middle approach to improving its behavior. LSTMs seem to be more flexible; this is likely due to deep neural networks' general function approximation ability. However, they take a long time to train and require a fine-tuning process. Overall, KNNRS could be used for a completely automatic regression with good performance,

regardless of the time series' nature. Perhaps a fully automatic monitor using KNNRS could be interleaved with LSTMs when an unusual event occurs or while hyper-parameters are optimized through a meta-learning procedure.

Acknowledgments

G. Pineda-Garcia is funded by the European Union's Horizon 2020 research and innovation program under the HBP SGA3 grant agreement.

References

[1] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence, Warwick 1980*, pp. 366–381, Springer, 1981.

[2] J. Theiler, "Estimating fractal dimension," *JOSA A*, vol. 7, no. 6, pp. 1055–1073, 1990.

[3] M. B. Kennel, R. Brown, and H. D. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Physical review A*, vol. 45, no. 6, p. 3403, 1992.

[4] T. Gautama, D. P. Mandic, and M. M. Van Hulle, "A differential entropy based method for determining the optimal embedding parameters of a signal," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 6, pp. VI–29, IEEE, 2003.

[5] M. Camplani and B. Cannas, "The role of the embedding dimension and time delay in time series forecasting," *IFAC Proceedings Volumes*, vol. 42, no. 7, pp. 316–320, 2009.

[6] D. Chelidze, "Reliable estimation of minimum embedding dimension through statistical analysis of nearest neighbors," *Journal of Computational and Nonlinear Dynamics*, vol. 12, no. 5, p. 051024, 2017.

[7] S. C. Wheelwright, S. G. Makridakis, *et al.*, *Forecasting methods for management*. Wiley, 1985.

[8] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software*, vol. 26, no. 3, pp. 1–22, 2008.

[9] H. Kantz and T. Schreiber, *Nonlinear time series analysis*, vol. 7. Cambridge university press, 2004.

[10] E. De La Vega, J. J. Flores, and M. Graff, "k-nearest-neighbor by differential evolution for time series forecasting," in *Nature-Inspired Computation and Machine Learning*, pp. 50–60, Springer, 2014.

[11] J. J. Flores, J. R. C. González, R. L. Farias, and F. Calderon, "Evolving nearest neighbor time series forecasters," *Soft Computing*, pp. 1–10, 2017.

[12] J. J. Flores, J. Ortiz, J. Cedeño González, C. Lara, and R. Farías, "Fnn a fuzzy version of the nearest neighbor time series forecasting technique," in *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pp. 1–6, 2015.

[13] J. Ortiz-Bejar, M. Graff, E. S. Tellez, J. Ortiz-Bejar, and J. C. Jacobo, "k-nearest neighbor regressors optimized by using random search," in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pp. 1–5, IEEE, 2018.

[14] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[15] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[18] O. Pupo-Roncallo, J. Campillo, D. Ingham, K. Hughes, and M. Pourkashanian, "Renewable energy production and demand dataset for the energy system of colombia," *Data in brief*, vol. 28, p. 105084, 2020.